Octave Extension Module

These materials were created as part of an outreach activity at NC State; here is the website containing the materials. The authors would like to acknowledge the support of the National Science Foundation through the grant DMS-1845406.

Activity 1 – Entering Matrices

1. First we will enter a vector, a one dimensional matrix, for now you will type commands at the command prompt that looks like ">"
   Type: a = [1 2 3 4]

You will see the elements displayed back to you in the next line.

2. Next we will enter a 3X3 matrix

   Type: b = [1 2 3; 4 5 6; 7 8 9]

You will see the matrix displayed back to you in columns and rows.
Each semicolon creates a new row.

3. We can perform mathematical calculations using this matrix. Try the following, and record the results:

ENTER	RESULT	EXPLAIN
2*a		
b + b		
b/2		
b**2		
a*a		

4. We can index certain values in a Matrix:

Try typing the following at the command prompt and explain what happens:

```
b(1,2)

b(:,2)

b(1,:)

b(1,2) + b(2,3)

sum(b)

max(b(3,:))
```

5. We have been entering matrices and testing commands from the command prompt. In order to save program code, open either the Octave editor or Notepad2 to add, edit, and save code. In the Octave editor or Notepad, save the file as a name.m file. Save it into the directory you are working in. To find the directory that you are currently in type "pwd" at the command prompt.

6. If the file is saved in the same working directory you can run any program by typing the name of the file (without the .m) at the command prompt.

For example:

A file is saved in the working directory and named matrix1.m

To run the file go to the command prompt and type matrix1 and hit enter

>>matrix1

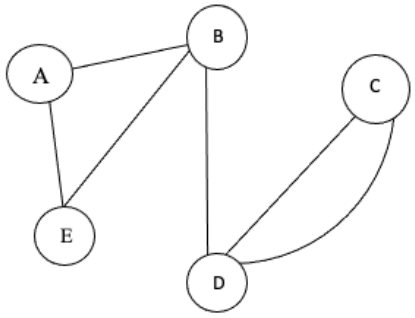
7. A few other helpful Octave commands

```
>> clear all          #clears all variables running at the command prompt

>> clc                #clears the screen, but keeps the variables that have been entered
```

Activity 2 – Vertex Edge Graphs and adjacency matrices

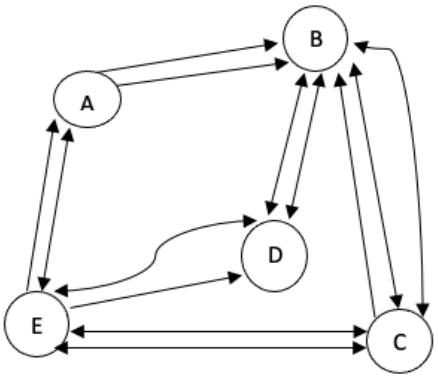
A **vertex edge graph** is made up of vertices that are connected by edges that symbolize a relation between nodes. The edges are undirected if the relationship extends both ways and directed (with arrows) if the relationship only extends in one direction.

Corresponding Adjacency Matrix for the given undirected graph:



	A	B	C	D	E
A	0	1	0	0	1
B	1	0	0	1	1
C	0	0	0	2	0
D	0	1	2	0	0
E	1	1	0	0	0

1) Look at vertex A in the graph above it has two links with B and E, respectively. Now look at the corresponding matrix and read row A. There is a 1 in the B and E columns to indicate that there is one edge from A to B and one edge from A to E. Since this graph is undirected the matrix is symmetric across the diagonal. For example the value in (A,B) is the same as (B,A). *Note: The degree of each vertex is the sum of each row or column. So vertex A has a degree of 2, B=3, C=2, D =3, and E=2.



	A	B	C	D	E
A	0	2	0	0	1
B	0	0	2	2	0
C	0	3	0	0	2
D	0	2	0	0	1
E	2	0	2	2	0

2) Look at the graph above. This graph is a directed graph so the matrix is not necessarily symmetric. Vertex A has 1 directed edge from A to E, but 2 directed edges from E to A. This is seen in the

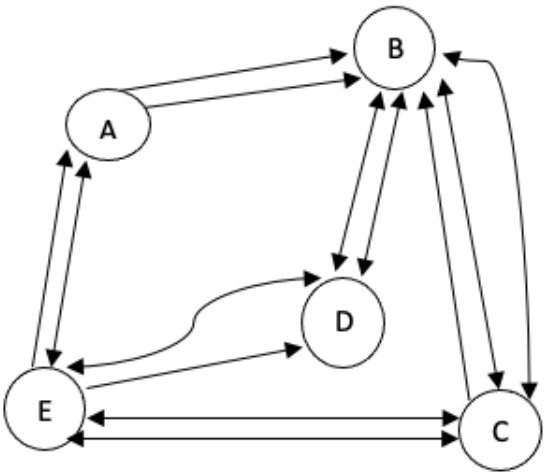
corresponding matrix. *Note: the sum of the rows is the **outdegree** of each vertex and the sum of the columns is the **indegree of each vertex.**

3) These are just small examples. Imagine having a city with hundreds or thousands of streets to analyze, we would want to automate the process. Spend some time exploring the Octave Code provided below. Experiment with creating matrices and graphs. Then try to find the number of nodes, edges, and degrees of the graphs you generate.

Useful Octave Code Chunks for Generating Matrices and Graphs:

```
%% create a random 10X10 matrix
m1 = round(2*rand(10,10))

%% take a random matrix and transpose to make it symmetric
m2 =max(m1,transpose(m1))


%% Define the graph through the adjacency matrix
A = [0 1 1 1 1 0 0 0; 1 0 0 0 0 0 0 0; 1 0 0 0 0 0 0 0; 1 0 0 0 0 0 0 0; 1 0 0 0 0 1 1 0; 0 0 0 0 1 0 1 0; 0 0 0 0 1 1
0 1; 0 0 0 0 0 0 1 0];


%% Investigate the statistics of this graph
d = sum(A,2);
disp('The number of nodes on this graph is')
length(d)
disp('The number of edges on this graph is')
nnz(triu(A))
disp('Average degree of this graph is')
mean(full(d))

%% Visualize the Distribution of the degree
figure, hist(d)
title('Distribution of the degree')
set(gca, 'FontSize', 16)

%% Cool formula to compute the number of triangles in the graph
disp('Number of triangles in this graph is')
trace(A^3)/6
```

Activity 3 – Reachability using Matrices



	A	B	C	D	E
A	0	2	0	0	1
B	0	0	2	2	0
C	0	3	0	0	2
D	0	2	0	0	1
E	2	0	2	2	0

A matrix can be used to store connections from one vertex to another, including connections that differ in different directions. For example, AE has one train connection, while EA has two train connections.

1) How many ways are there to travel from E B with exactly one stop?

2) Did you use the graph or the matrix?

3) There are three routes: EAB and EDB and ECB

4) Using the matrix, multiply and add the elements: $M_{51}*M_{12}$ + $M_{54}*M_{42}$ + $M_{53}*M_{32}$

5) Is the result the same as your answer from #1? If not, it's easy to miss a route.

Imagine a network system with hundreds of cities, thousands of train routes. It would be very difficult to handle this information without a matrix structure.

6) Open a new file in the Octave editor or Notepad2. Create a new random matrix by typing:

T=round(4*rand(7,7))

7) Save this file as Train. Open the command window and type: Train

8) The matrix should display numbers between 0 and 5 and is not symmetric. It shouldn't be since train routes may vary by direction.

9) Let's say this matrix represents the train routes between 7 cities labeled A-G. Now we will index the values we need to find the number of ways to travel from EB, as illustrated above except this time we must also make a stop in F and G. Type:

```
trains = round(4*rand(7,7));                          #creates a random matrix
stops = [1 2 3 4 5 6 7];                              #index numbers for stops A-G
total = 0;                                                    #variable for total set to zero
for i = 1:length(stops)                                  #for loop to iterate through all
stops
    routes = trains(5,stops(i))*trains(stops(i),2);   #index calculations
    total +=routes;                                          #adds routes to the total
endfor                                                        #ends the for loop
total                                                          #returns the total routes for EB
```

10) Examine the code and try to determine what is happening. The comments on the right are to explain each step. To see more of what is happening, delete the semicolons at the end of lines and rerun the program.

11) This small program works well, but it only finds the number of routes from EB, and would have to be rewritten for another scenario.

12) Add the following lines to your program:

```
result = trains**2
result(5,2)
```

13)Rerun the program. The answer for total and result should be the same. Try running it a few more times.

14) By simply squaring the matrix and indexing the result for City E to City B, we have the correct number. This is a matrix shortcut to solving the total possible connections between two points.

15) This works with any number of connections. For one stop we use the power of 2 since there are two legs to the journey. For two stop journeys we would cube the matrix, since the trip would have three legs.

16) Open a new file and write this simpler solution. Name in train2

a) create a random 10x10 matrix
b)cube the matrix
c) index the value that would be the number of routes from City 3 to City 8

17) Finally, try a much larger matrix, with dimensions of 100 or 1000. **IMPORTANT:** remember to suppress the output using a semicolon at the end of the first 2 lines. Numbers larger than 1000 will take longer.

Challenge:

- Reduce your trains matrix back to a 7x7 matrix.
- Create another random matrix for bus routes between the cities:
- You are traveling from City 3 to City 8 by train, then back from City 8 to City 3 by bus

a) Write the code to find one stop on the train route and one on the bus route?

b) Write the code to find two stops on the bus and one on the train?

c) How could you change it to find the number of routes taking either bus or train with 3 stops?

Note: This example also works in another interesting way, Social Networking. A single matrix would represent direct friendships, a squared matrix represents friends of friends and a cubed matrix is friends of friends of friends.

Activity 4: Friendship Paradox

A paradox is "a statement or proposition that seems self-contradictory or absurd but in reality expresses a possible truth." The friendship paradox implies on average your friends have more friends than you. Stated more precisely, on average people have more friends of friends than they do friends.

Example: Consider 4 people (nodes A, B, C, and D) and their friendships (indicated by edges) represented in the following graph:

What is the average number of friends among these 4 people? How do we calculate it?

Person	A	B	C	D
Number of friends	1	3	2	2

Average number of friends = $\overline{F} = \frac{Total\ number\ of\ friends}{Number\ of\ people} = \frac{1+3+2+2}{4} = 2.$

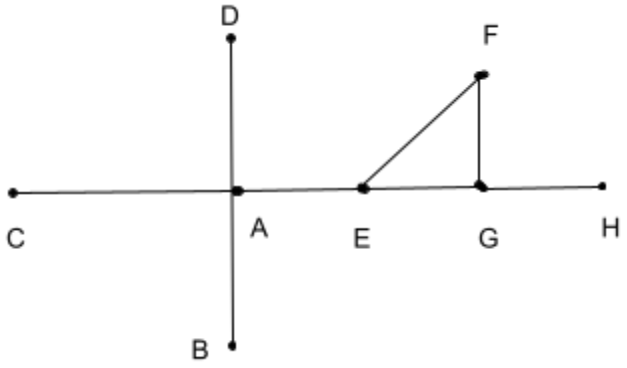
Person	Number of friends	Number of friends of friends
A	1 (B)	3
B	3 (A, C, D)	5
C	2 (B, D)	5
D	2 (C, B)	5

Average number of friends of friend:

$$F = \frac{Total\ number\ of\ friends\ of\ friends}{Total\ number\ of\ friends} = \frac{3+5+5+5}{1+3+2+2} = \frac{18}{8} = 2.25$$

Although the math is simple, it is computationally heavy. This is a lot of work for a very small network of 4 friends. This is where computers come in handy. Let's take a look at another small network and use Octave to help us solve it.

Exercise: Use the following graph to calculate $\overline{F}$ and $\overline{FOF}$.



1) Open Octave and at the command prompt define an adjacency matrix that matches the graph above. See the code provided below:

%% Define the graph through the adjacency matrix
A = [0 1 1 1 1 0 0 0; 1 0 0 0 0 0 0 0; 1 0 0 0 0 0 0 0; 1 0 0 0 0 0 0 0; 1 0 0 0 0 1 1 0; 0 0 0 0 1 0 1 0; 0 0 0 0 1 1 0 1; 0 0 0 0 0 0 1 0];

2) Once you have matrix A. At the command prompt use the sample code below to find the following:

Number of Nodes	Number of Edges	Average Degree of the Graph

%% Investigate the statistics of this graph
d = sum(A,2);
disp('The number of nodes on this graph is')
length(d)
disp('The number of edges on this graph is')
nnz(triu(A))
disp('Average degree of this graph is')
mean(full(d))

3) At the command prompt use the sample code below to find the following:

Average Number of Friends	Average Number of Friends of Friends

```
%% Explore the Friendship paradox
disp('Average number of friends')
d = sum(A,1);
f_bar = mean(d)


% Method 1: Compute friends of friends
for j = 1:length(d)
   friends = find(A(:,j)); %Nonzeros correspond to friends
   num_friends(j) = length(friends);
   num_friends_of_friends(j) = sum(sum(A(:,friends)));
end
disp('Number of friends and number of friends of friends')
[num_friends' num_friends_of_friends']
disp('Average number of friends of friends using Method 1')
fof_bar = sum(num_friends_of_friends)/sum(d)


% Method 2: Using the formula
disp('Average number of friends of friends using Method 2')
fof_bar = sum(d.^2)/sum(d)
```

4) Challenge: Draw your own random network graph of friends below. Use as many vertices as you want, but remember you will have to create and enter the adjacency matrix into Octave.

Enter the matrix into Octave and use the same code to find the following:

Number of Friends:

Number of Friends of Friends:

Reflection: Did the Friendship Paradox hold true? Do you think it always holds true? Why or why not?

5) Extension with a very large network.

This extension is for students to be able to visualize the connections between a very large network. Students will need to be able to download the file powergrid.edgelist.txt (from this website) and save it to the directory they are working in. This network represents the Western States Power Grid. Each node is a power plant, transformer or consumer, and two nodes are connected if they are physically connected via a cable.

Ref: Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of ā??small-worldā?? networks. nature, 393(6684), 440-442.

★ This code is to read the data set. Make sure the file, powergrid.edgelist.txt, is saved in the same directory that Octave is working in.

```
%% Read the dataset
[I,J] = textread('powergrid.edgelist.txt');
M = [I,J];
At = sparse(M(:,1)+1, M(:,2)+1, ones(size(M,1),1), max(max(M))+1, max(max(M))+1);
A = At+ At'; % Symmetrize it for an undirected graph
```

★ The following code is to visualize part of the graph. When examining the graph, notice the dense blocks near the diagonals. These are indicative of clusters of densely connected nodes.

```
%% Visualize part of the graph
figure, spy(A);
title('Sparsity pattern of the graph')
set(gca, 'FontSize', 16)
```

★ Use the code below to find the number of nodes and edges as well as the average degree of the graph. Remember that the degree is the number of edges connected to a node. The histogram will display a distribution of the degrees of each vertex. Notice that even though the network is vast, the degrees of most individual vertices are relatively small.

```
%% Investigate the statistics of this graph
d = sum(A,2);
disp('The number of nodes on this graph is')
length(d)
disp('The number of edges on this graph is')
nnz(At)
disp('Average degree of this graph is')
mean(full(d))

% Visualize the Distribution of the degree
figure, hist(d)
title('Distribution of the degree')
set(gca, 'FontSize', 16)
.
```